Collaborative Meta-Learning

Joaquin Vanschoren [1] and **Larisa Soldatova** [2]

Abstract. This paper proposes an ontology and a markup language to describe machine learning experiments in a standardized fashion and support a *collaborative* approach to the analysis of learning algorithms. Experiments can then be shared with the community and augmented with large amounts of experiments by other researchers as well as all measurable properties of algorithms and datasets. By maximally reusing what has been learned before, by many researchers, this approach enables machine learning researchers to perform in-depth meta-learning studies with minimal effort, e.g., to investigate and explain algorithm performance on different types of data and under different parameter settings. As can been learned from recent developments in other sciences, such a free exchange and reuse of experiments requires a clear representation. We therefore focus on the design of ontologies and formal representation languages to express and share machine learning meta-data with the world.

1 Motivation

1.1 Experimentation in machine learning

Research in machine learning is inherently empirical. Researchers, as well as practitioners, seek a deeper understanding of learning algorithm performance by performing large numbers of learning experiments. Whether the goal is to develop better learning algorithms or to select useful approaches to analyze new sources of data, collecting the right meta-data and correctly interpreting it is crucial for a thorough understanding of learning processes.

However, running those experiments tends to be quite laborious. In the case of evaluating a new algorithm, pictured in Figure 1, one needs to search for datasets, preprocessing algorithms, (rival) learning algorithm implementations and scripts for algorithm performance estimation (e.g. cross-validation). Next, one needs to set up a wide range of experiments: datasets need to be preprocessed and algorithm parameters need to be varied, each of which requires much expertise, and experiment designs [17] need to be employed to thoroughly test the influence of different experimental variables. Finally, after all experiments have run, one needs to properly organize all the collected results in order to interpret them correctly. This easily amounts to a large range of experiments representing days, if not weeks of work, while only averaged results will ever be published. Any other researcher willing to verify some results or test a certain hypothesis will have to start again from scratch, repeating the same experiments instead of simply reusing them.

1.2 Generalizability and Interpretability

Moreover, in order to ensure that results are generally valid, the empirical evaluation also needs to be equally general, meaning that it must cover many different conditions. These include various parameter settings and various kinds of datasets, e.g. differing in size, skewness, noisiness or with or without being preprocessed with basic techniques such as feature selection. Unfortunately, because of the amount of work involved in empirical evaluation, many studies will not explore these conditions thoroughly, limiting themselves to testing algorithms (often only with default parameter settings) on a selection of benchmark datasets. It has long been recognized that such studies are in fact only 'case studies' [1], and should be interpreted with caution.

Indeed, sometimes, overly general conclusions can be drawn. First, in time series analysis research, many studies were shown to be biased toward the datasets being used, leading to contradictory results [16]. Second, it has been shown that the relative performance of learning algorithms can depend heavily on the amount of sampled training data (their learning curves cross) [21, 28], and is also easily dominated by the effect of parameter optimization and feature selection [13]. Since only few studies thoroughly vary data sample sizes, parameter settings or feature sets, it is not clear how generally valid their results are.

As such, there are good reasons to thoroughly explore different conditions, or at least to clearly state under which conditions certain conclusions may or may not hold. Otherwise, it is very hard for other researchers to correctly interpret the results, thus possibly creating a false sense of progress [10]:

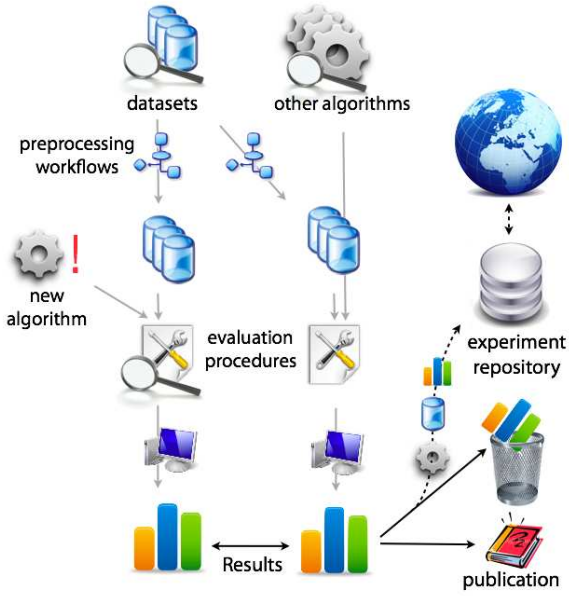


Figure 1. A typical experiment workflow in machine learning research.

[1] K.U. Leuven, Belgium, email: joaquin.vanschoren@cs.kuleuven.be
[2] Aberystwyth University, UK, email: lss@aber.ac.uk

...no method will be universally superior to other methods: relative superiority will depend on the type of data used in the comparisons, the particular data sets used, the performance criterion and a host of other factors. [...] an apparent superiority in classification accuracy, obtained in laboratory conditions, may not translate to a superiority in real-world conditions...

1.3 Large-scale studies

Several comprehensive empirical studies exist that try, as well as possible, to cover a wide range of conditions. The StatLog [18] and MetaL [5] projects and, more recently, some large empirical studies, for instance Ali and Smith [2] and Caruana and Niculescu [6], aim to extract very general conclusions from large and very general experiment setups. Still, as new algorithms, preprocessing methods, learning tasks, and evaluation metrics are introduced at a constant rate, it is impossible for a single study to cover this continuously expanding space of learning approaches. Moreover, the meta-data generated by these and thousands of other machine learning studies is usually collected and stored differently and therefore hard to share and reuse.

2 A collaborative approach

In this paper, we advocate a much more dynamic, *collaborative* approach to experimentation, in which experiments can be freely shared, linked together, augmented with measurable properties of algorithms and datasets, and immediately reused by researchers all over the world. Any researcher creating empirical meta-data should thus be able to easily share it with others and in turn reuse any prior results of interest. Indeed, by reusing prior results we can avoid unnecessary repetition and speed up scientific research, and by bringing the results of many studies together, we can obtain an increasingly detailed picture of learning algorithm behavior. In turn, this facilitates large-scale, very generalizable machine learning studies which are prohibitively expensive to start from scratch.

2.1 e-Sciences

The use of such public experiment repositories is common practice in many other scientific disciplines, such as micro-array repositories in bio-informatics [25] and virtual observatories in astrophysics [26]. These so-called e-Sciences aim to share as much empirical data as possible online, creating an "open scientific culture where as much information as possible is moved out of people's heads and labs, onto the network and into tools that can help us structure and filter the information" [19].

Ironically, while machine learning and data mining have been very successful in speeding up scientific progress in these fields by discovering useful patterns in a myriad of collected experiment results, machine learning experiments themselves are currently not being documented and organized well enough to engender the same automatic discovery of insightful patterns that may speed up the design of new algorithms or the selection of the best algorithms to analyze new collections of data.

2.2 An infrastructure for experiment exchange

Similar to developments in these e-Sciences, we need to clearly formalize machine learning techniques and investigations, and set up

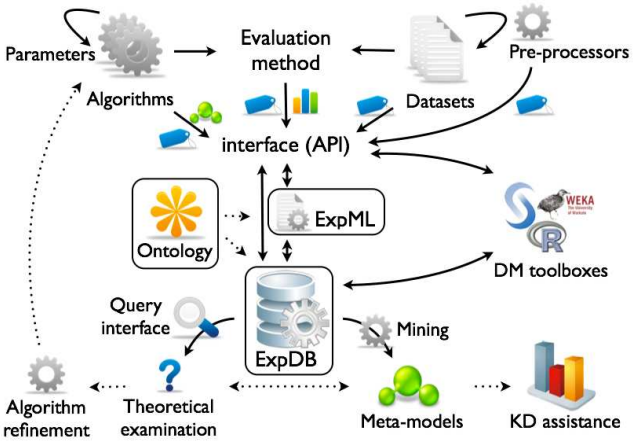


Figure 2. An infrastructure for experiment exchange.

online infrastructures for experiment exchange. The latter typically consist of more or less the same three components, shown in the boxed elements of Figure 2:

A formal representation language. To enable a free exchange of experiment data, a standard and formal representation language needs to be agreed upon. Such a language may also contain guidelines about the information necessary to ensure reproducibility.

Ontologies. Defining a coherent and unambiguous description language is not straightforward. It requires a careful analysis of all the concepts used within a domain and their relationships. The result can be formally represented in *ontologies* [7]: machine manipulable domain models in which each concept is clearly described. They establish a common vocabulary for describing experiments, so that experiments by other researchers can be clearly interpreted, even by software agents.

A searchable repository. To reuse experiment data, we need to locate it first. Experiment repositories therefore still need to organize all data to make it easily retrievable. Querying tools or query languages can be used to facilitate access.

To realize this system, we build upon *experiment databases* (ExpDBs) [3, 28, 27]: databases designed to collect the details of these experiments, and to intelligently organize them in online repositories to enable fast and thorough analysis of a myriad of collected results. While the design and use of experiment databases in machine learning has been amply discussed before, this paper will focus on the design of the remaining two components needed to extend and use them collaboratively. We will first introduce Exposé, a proposed ontology for machine learning experimentation, and next, we will illustrate how we can translate this ontology into a formal, XML-based representation language called ExpML.

To make the sharing of experiments as easy as possible, experiment descriptions should be generated automatically: a programming interface (API), shown in the top center of Figure 2 is provided that allows to build uniform representations of algorithm implementations, datasets and entire experiments. This interface can be included in data mining tools so that experiments can be shared (or downloaded) at the click of a button. Measurable properties of algorithms and datasets (shown as labels in Figure 2) can be added as well. Next, the experiments are automatically exported to a common format (ExpML) and organized in experiment databases. Such databases can be setup for personal use, to organize all of one's experiments, or to build lab-wide or world-wide repositories.

The arrows emanating from the ExpDB at the bottom of Figure 2 shows different ways to tap into the shared meta-data:

Querying allows a researcher to formulate questions about the stored experiments, and immediately get all results of interest. Several query interfaces have been developed, both online and in a downloadable tool, available on `http://expdb.cs.kuleuven.be`. They include a graphical query composer helping the user to quickly select experiment details or impose constraints, and offer various visualizations of the returned results.

Mining. A second use is to automatically look for patterns in algorithm performance by mining the stored meta-data. The insights provided by such *meta-models* can then be used to design better algorithms or to select and apply them in knowledge discovery applications [5].

Integration. Data mining toolboxes could also interface with ExpDBs directly. All experiments performed with the toolbox could be automatically exported to ExpDBs, and previous experiments, run before by a different user of that toolbox, can be downloaded to avoid repetition and speed up experimentation.

In the remainder of this paper, we will first outline Exposé, our proposed ontology, in Section 3, and the resulting XML-based language, ExpML, in Section 4. Finally, we provide some illustrations of possible meta-learning studies in Section 6. Section 7 concludes.

3 The Exposé ontology

Sharing machine learning experiments with each other starts with speaking the same language. Moreover, if we want to automatically organize thousands of experiments generated by various researchers all over the world, this language should be interpretable by machines. Designing a coherent and unambiguous formal language is not straightforward, especially since experimentation in machine learning is a very involved process including various statistical techniques and many different setups. Indeed, a very fine-grained description will be needed if we wish to answer questions about detailed aspects of the involved learning algorithms and datasets.

In this section, we briefly describe an ontology [7, 12], called Exposé, in which the concepts used within machine learning experiments and their relationships are carefully described and expressed in a formal domain model. It provides a common, unambiguous core vocabulary for researchers wishing to describe their experiments, and it explicates the inherent structure of machine learning experiments. Ontologies are a logical choice for the principled design of community-based experiment databases, since they are built to evolve: they can be modified, extended and refined by many different researchers to cover ever more types of experiments, tasks and algorithms. We can thus edit or extend our domain model on a conceptual level, and 'translate' these changes into updated markup languages and database models, so that they also evolve with the field.

In a way, we start small. We will focus on supervised classification and our experiments are limited to algorithm evaluations on static, propositional datasets. Still, this already covers a decent amount of contemporary experimentation and includes many concepts common to other subfields. It is important to note that this is a straw-man proposal that is intended to instigate discussion and attract wider involvement from the data mining community. It has been influenced and adapted by many people, mostly through close collaboration with the authors of other data mining ontologies.

Exposé is described in the OWL-DL ontology language [12]. It can be downloaded from the experiment database website (`http://expdb.cs.kuleuven.be`), and explored and edited using any OWL-DL editor, e.g. the Protégé editor (v4)[3].

In all, the ontology currently defines over 850 classes (concepts), most of which describe algorithms, dataset and algorithm characteristics, experiment design methods and evaluation functions.

3.1 Ontology design

In designing Exposé, we paid close attention to existing guidelines for ontology design [15]:

Top-level ontologies. It is considered good practice to start from generally accepted and unambiguously described classes (concepts) and properties (relationships) [20]. We started from the Basic Formal Ontology (BFO)[4] covering top-level scientific classes and the OBO Relational Ontology (RO)[5] offering a predefined set of relationships.

Ontology reuse. If possible, (parts of) other ontologies should be reused to build on the knowledge (and the consensus) expressed in those ontologies. When designing Exposé, we reused general machine learning related classes from the OntoDM ontology [20] (a general, high-level ontology for data mining with the aim of providing a unified framework for data mining research), experimentation-related classes from the EXPO ontology [24] (a top-level ontology for scientific experiments containing classes for hypotheses, (un)controlled variables, experiment designs and experiment equipment), and classes related to internal algorithm mechanisms from the DMOP ontology [11]. In fact, Exposé bridges the gap between the very specific classes of DMOP and the very general ones of OntoDM, thus providing an important contribution to the harmonization of various data mining ontologies. Any future extensions of any of these other ontologies can directly be used to update Exposé, and vice-versa.

Design patterns. Ontology design patterns[6] are reusable, successful solutions to recurrent modeling problems. For instance, we mentioned that a learning algorithm can act as an individual learner in one experiment, and as a base-learner for an ensemble learner in the next. This is a case of an agent-role pattern, in which an agent (algorithm) only plays a certain role in a process in some occasions, but not always. A predefined relationship, 'realizes', is used to indicate that an individual is able to fulfill a certain role. We have used such patterns as often as we could.

Quality criteria. General criteria include *clarity* (descriptions of the classes should make the meaning of each concept clear), *coherence* or *consistency* (there should be no logical contradictions), *extensibility* (future uses should be anticipated) and *minimal commitment* (only support the intended knowledge sharing activities). These criteria are rather qualitative, and were only evaluated through discussions with other researchers.

Many classes and properties were extracted from earlier working versions of our experiment database [3], and thus proved practically useful to organize experiment information. Vice versa, many limitations of those earlier versions were solved through Exposé's much more principled domain model.

[3] http://protege.stanford.edu/
[4] http://www.ifomis.org/bfo
[5] http://www.obofoundry.org/ro/
[6] http://ontologydesignpatterns.org

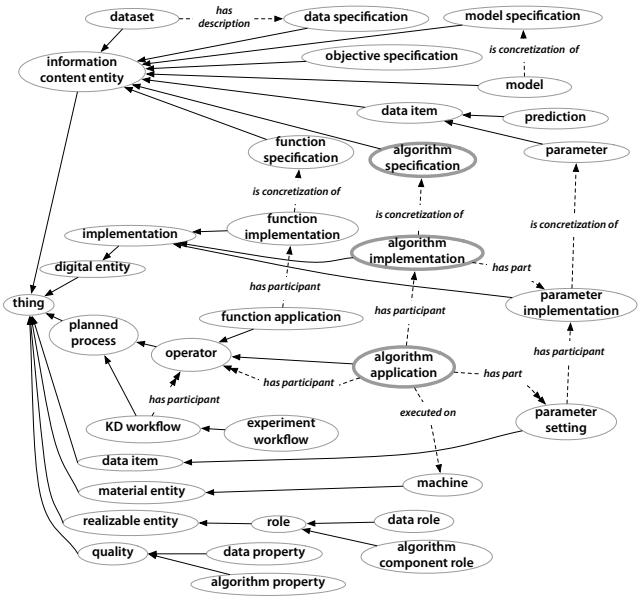


Figure 3. An overview of the top-level classes in the Exposé ontology.

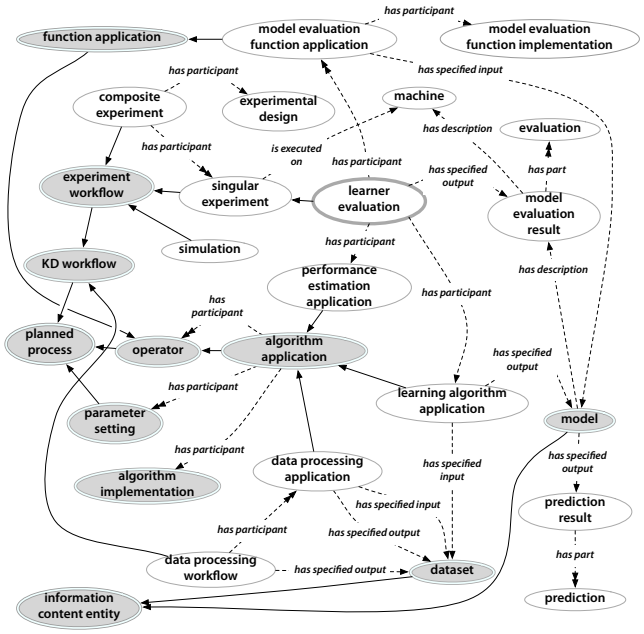


Figure 4. Experiments in the Exposé ontology.

3.2 Top-level View

Figure 3 shows the most important top-level classes and properties, many of which are inherited from the OntoDM ontology [20], which in turn reuses classes from OBI[7] (i.e., planned process) and IAO[8] (i.e. information content entity). The full arrows symbolize an 'is-a' relationship, meaning that the first class is a subclass of the second, and the dashed arrows symbolize other common relationships. Double arrows indicate one-to-many relationships, for instance, an *algorithm application* can have many *parameter settings*.

The three most important categories of classes are *information content entity*, which covers datasets, models and abstract specifications of objects (e.g. algorithms), *implementation*, and *planned process*, a sequence of actions meant to achieve a certain goal. When describing experiments, this distinction is very important. For instance, the class 'C4.5' can mean the abstract algorithm, a specific implementation or an execution of that algorithm with specific parameter settings, and we want to distinguish between all three.

As such, ambiguous classes such as 'learning algorithm' are broken up according to different interpretations (indicated by bold ellipses in Figure 3): an abstract *algorithm specification* (e.g. in pseudo-code), a concrete *algorithm implementation*, code in a certain programming language with a version number, and a specific *algorithm application*, a deterministic function with fixed parameter settings, run on a specific *machine* with an actual input (a *dataset*) and output (a *model*), also see Figure 4. The same distinction is used for other algorithms (for data preprocessing, evaluation or model refinement), mathematical functions (e.g. the kernel used in an SVM), and parameters, which can have different names in different implementations and different value settings in different applications. Algorithm and function applications are *operators* in a workflow (see below), and can even be participants of another algorithm application (e.g., a kernel or a base-learner), i.e. they can be part of the inner workflow of an algorithm.

Finally, there are also *qualities*, properties of a specific dataset or algorithm (see Figures 6 and 7), and *roles* indicating that an element assumes a (temporary) role in another process: an algorithm act as a base-learner in an ensemble, a function can act as a distance function in a learning algorithm, and a dataset can be a training set in one experiment and a test set in the next.

[7] http://obi-ontology.org
[8] http://code.google.com/p/information-artifact-ontology

3.3 Experiments

Figure 4 shows the ontological description of experiments, with the top-level classes from Figure 3 drawn in filled double ellipses. Experiments are defined as *workflows*. The general nature of workflows allows the description of many kinds of experiments. Some (composite) experiments can also consist of many smaller (singular) experiments, and can use a particular *experiment design* [17] to investigate the effects of various *experimental variables*, e.g. parameter settings.

We will now focus on a particular kind of experiment: a *learner evaluation* (indicated by a bold ellipse). This type of experiment applies a specific learning algorithm (with fixed parameters) on a specific input dataset and evaluates the produced model by applying one or several model evaluation functions, e.g. predictive accuracy. In predictive tasks, a performance estimation technique, e.g. 10-fold cross-validation, is applied to generate training- and test sets, evaluate the resulting models and aggregate the results. After it is executed on a specific *machine*, it will output a *model evaluation result* containing the outcomes of all evaluations and, in the case of predictive algorithms, the (probabilistic) predictions made by the models. Models are also generated by applying the learning algorithm on the entire dataset.

Finally, more often than not, the dataset will have to be preprocessed first. Using workflows, we can define how various *data processing applications* preprocess the data before it is passed on to the learning algorithm. Figure 5 illustrates such a workflow. The top of the figure shows that it consists of participants (operators), which in turn have inputs and outputs (shown in ovals): datasets, models and model evaluation results. Workflows themselves also have inputs and outputs, and we can define specialized sub-workflows. A *data processing workflow* is a sequence of data processing steps. The center of Figure 5 shows one with three preprocessors. A *learner evaluation*

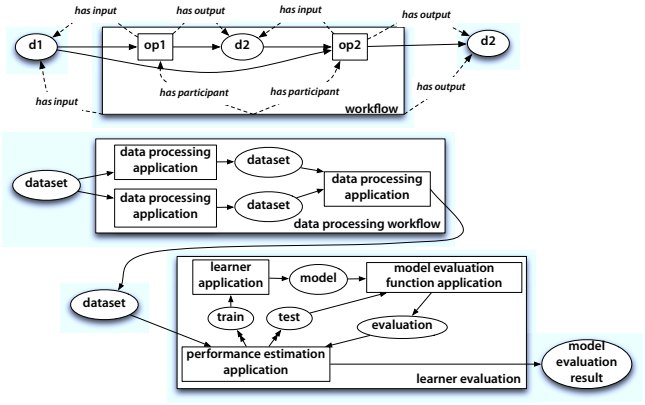


Figure 5. Experiment workflows.

workflow takes a dataset as input and applies performance estimation techniques (e.g. 10-fold cross-validation) and model evaluation functions (e.g. the area under the ROC curve) to evaluate a specific learner application.

This constitutes a clear definition of how the learning algorithm is evaluated, making sure it can be interpreted unambiguously. Of course, there are other types of learner evaluations, both finer ones, e.g. a singular train-test experiment, and more complex ones, e.g. doing an internal model selection to find the optimal parameter settings. Exposé also defines 7 evaluation techniques and around 50 different evaluation functions, which can in turn have different implementations.

3.4 Datasets

Figure 6 shows the most important classes used to describe datasets.

Specification. The *data specification* (in the top part of Figure 6) describes the structure of a dataset. Some subclasses are graphs, sequences and sets of instances. The latter can have instances of various types, for instance tuples, in which case it can have a number of *data features* and *data instances*. For other types of data (e.g. relational data) this specification will have to be extended. Finally, a dataset has descriptions, including the dataset name, version, download url and a textual description of its origin, which are needed to make the dataset easily retrievable.

Roles. A specific dataset can play different *roles* in different experiments (top of Figure 6). For instance, it can be a training set in one evaluation and a test set in the next.

Data properties. As said before, we wish to link all empirical results to theoretical metadata, called *properties*, about the underlying datasets to perform meta-learning studies. These *data properties* are shown in the bottom half of Figure 6, and may concern individual instances, individual features or the entire dataset. We define both *feature properties* such as feature skewness or mutual information with the target feature, as well as general *dataset properties* such as the number of attributes and landmarkers [22].

3.5 Algorithms

One last aspect of the ontology we wish to highlight is the description of algorithms. To allow the exploration of algorithm performance under different configurations and parameter settings, and include all aspects that influence their performance in queries, we need a detailed vocabulary to describe them.

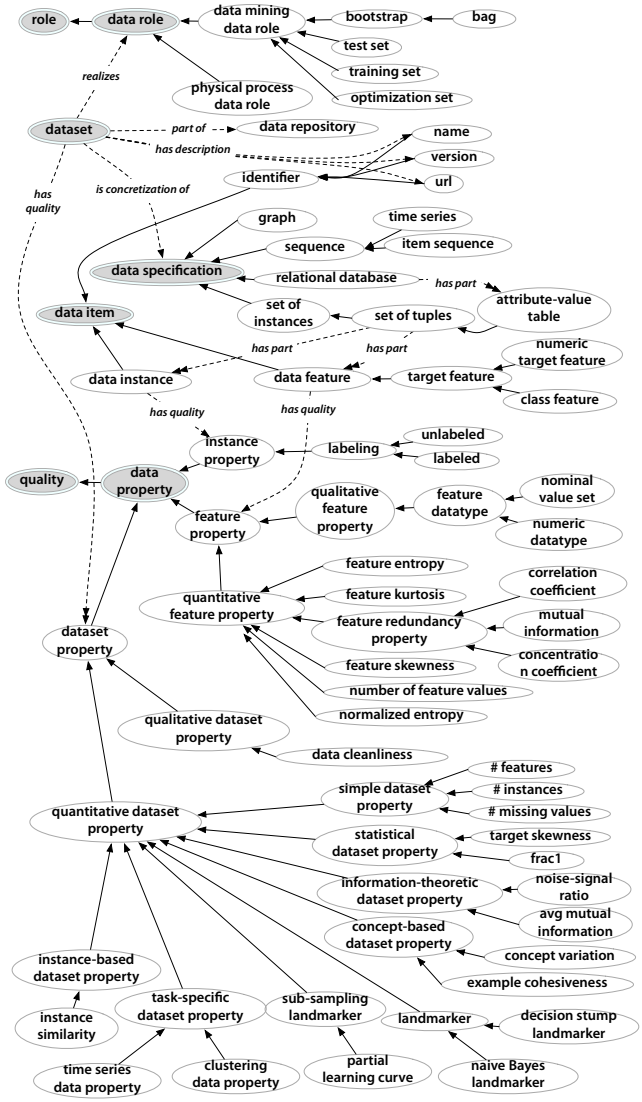


Figure 6. Datasets in the Exposé ontology.

Algorithm implementations. Figure 7 shows how algorithms and their configurations are expressed in our ontology. As mentioned in Section 3.2, algorithms can have various *implementations* and *applications* with specific *parameter settings*. The latter attach a value to a *parameter implementation*, which depending on the algorithm in which they are implemented, can have different names and different default values. Algorithm implementations are described with all information needed to retrieve and use them, such as their name, version, url, and the library they belong to (if any). Moreover, they can have *qualities*, e.g. their susceptibility to noise.

Algorithm composition. Some algorithms also use other algorithms or mathematical functions, which can often be selected (or plugged in) by the user. These include base-learners in ensemble learners, distance functions in clustering and nearest neighbor algorithms and kernels in kernel-based learning algorithms. Some algorithm implementations also use internal data processing algorithms, e.g. to remove missing values. In Exposé, any operator can be a participant of an algorithm application, combined in internal workflows with in- and outputs. Depending on the algorithm, operators can ful-

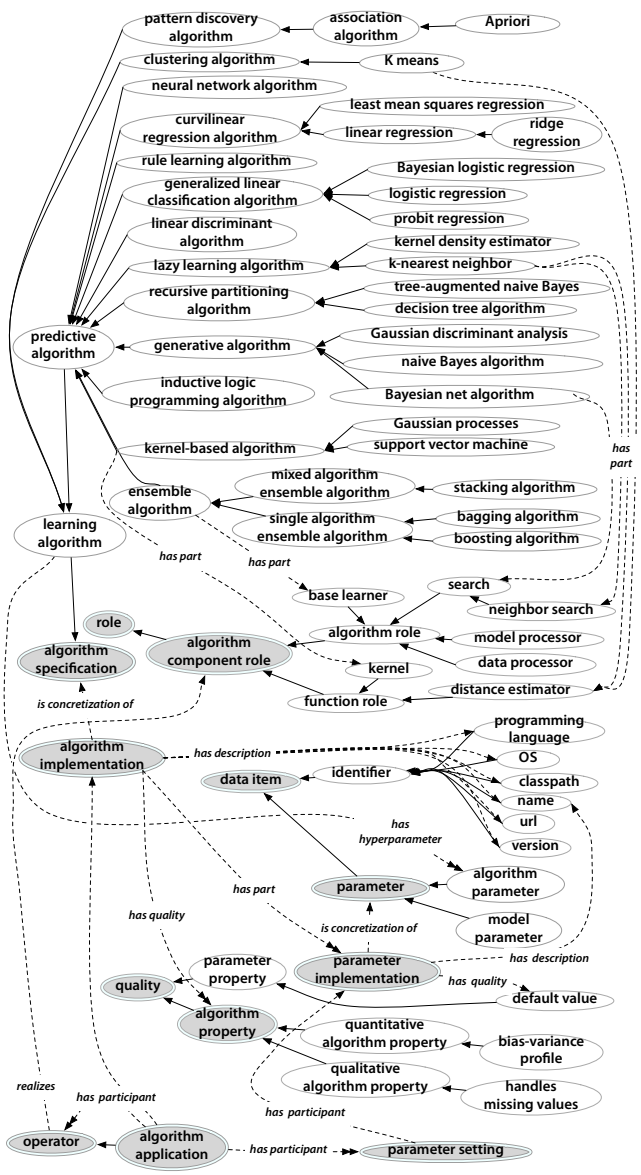


Figure 7. Algorithms and their configurations in the Exposé ontology.

fill (realize) certain predefined *roles* (center of Figure 7).

The top half of Figure 7 shows the classes of algorithms that are currently defined in the ontology, and which internal operators they have. For instance, a support vector machine (SVM) is always assumed to have a kernel, and the choice of kernel and its implementation will be fixed on application.

Algorithm mechanisms. Finally, to understand the performance differences between different types of algorithms, we need to look at the internal learning mechanisms on which they are built. These include the kind of models that are built (e.g. decision trees), how these models are optimized (e.g. the heuristic used, such as information gain) and the decision boundaries that are generated (e.g. axis-parallel, piecewise linear ones in the case of non-oblique decision trees). These classes, which extend the algorithm definitions through specific relationships (e.g. *has model structure*), are defined in the DMOP ontology [11], so they won't be repeated here.

4 The ExpML language

Using the Exposé ontology as our core vocabulary, we can define a formal markup language for describing experiments. This entails a translation of ontological classes and properties to XML elements and syntax. This translation process is especially useful because it allows ontological extensions (e.g. to new machine learning tasks) to be translated into updated ExpML definitions.

4.1 Operators and processes

First, we need to define which aspects of machine learning experiments we wish to share. We can divide these in two groups:

- Definitions of new experiment elements, such as new algorithms, datasets, evaluation functions and kernels. These correspond to algorithm or function *implementations* and *datasets* in our ontology.
- The experiment setups and results, corresponding to *planned processes*: *experiment workflows*, *data processing workflows*, and their in- and outputs, such as *model evaluation results*.

Because ontological relationships are more expressive than XML syntax, different relationships between these classes need to be translated quite differently. Table 1 provides a short overview of these relationships and their XML equivalent. Figures 8-10 illustrate this process, showing a real experiment (experiment 445080 in our experiment database) expressed in ExpML. We first describe a new algorithm implementation, and then we perform an experiment in which a dataset is preprocessed with a feature selection technique, and subsequently used to evaluate the added algorithm implementation.

4.2 New Operators

We start by describing a new algorithm implementation, i.e. WEKA's implementation of the bagging algorithm. Figure 8 shows the ExpML description, together with the corresponding Exposé classes above. First, we take the core class, *learning algorithm implementation*, and convert it into an XML element: *learner implementation*. Next, we express all its properties, as inherited from its parent, *algorithm implementation*.

4.3 Experiment Workflows

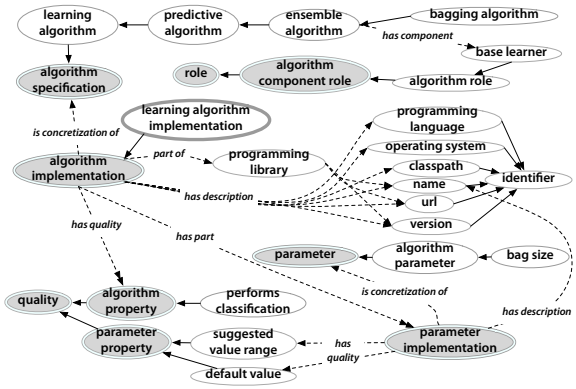
Figure 9 shows, from bottom to top, a simplified experiment workflow, the expression of the first half of this workflow in ExpML and the corresponding part of the ontology. To express workflows in ExpML, we assign an id to each operator and in- or output. For each operator, we state its inputs in an `input data` attribute, and for each output, we state the operator that generated it in an `output of` attribute. As shown in the ExpML code, a dataset `d1` is used as the input of workflow `op1` and data processing operator `op2`. The resulting dataset `d2` is referenced as both the output of operator `op1` and workflow `op2`.

The data processing workflow (Figure 9) contains a single participant: a data processing application, i.e. a feature selection algorithm. The ontology shows that this algorithm, in turn, requires an input, a dataset, and has a participant: a specific implementation. Since it is also an algorithm application, it can have multiple *parameter settings* and internal *operators*.

In this case, there are two operators: a feature subset evaluation function and a search algorithm, each *realizing* a certain *algorithm component role*. The first is realized by a *function application*,

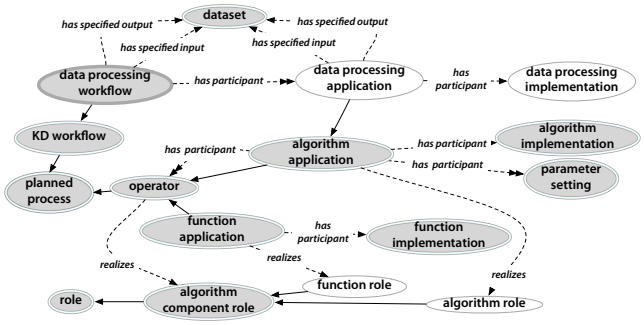
Table 1. Translating ontological properties to XML syntax.

Ontological property	XML syntax	example
has-part, pas-participant with role attribute if defined	target: subelement of source `parameter_impl` (Figure 8)	
has-description	(required) attribute	`name` attribute (Figure 8)
has-quality	subelement called `property`	`property` (Figure 8)
is-concretization-of	`implementation_of` attribute	`implementation_of` attribute (Figure 8)
part-of	specific attributes	`libname` and `libversion` (Figure 8)
has-specified-input	input given id, referenced in `input_data` attribute	`input_data='d1'` (Figure 9)
has-specified-output	source given id, referenced in `output_of` attribute	`output_of='d1'` (Figure 9)




```
<expml>
 <learner_impl name="weka.Bagging" version="1.31" os="any"
    url="http://..." language="java" libname="weka"
    implementation_of="bagging_algorithm"
    classpath="weka.classifiers.meta.Bagging">
  <parameter_impl name="P" implementation_of="bag_size">
    <property name="default_value">100</property>
    <property name="suggested_value_range">
        20,40,60,70,80,90,100</property>
  </parameter_impl>
  <parameter_impl name="I" implementation_of="
      number_of_iterations">
    <property name="default_value">10</property>
    <property name="suggested_value_range">
        10,20,40,80,160</property>
  </parameter_impl>
  <parameter_impl name="S" implementation_of="random_seed"
      >
    <property name="default_value">1</property>
  </parameter_impl>
  <learner_impl role="base-learner"/>
  <property name="performs_classification">true</property>
  <property name="handles_missing_values">true</property>
  ...
 </learner_impl>
```

Figure 8. An algorithm implementation, in Exposé and ExpML.




```
<dataset id="d1" name="kr-vs-kp" url="http://...">
 <target_feature name="class" index="-1" />
</dataset>
<data_processing_workflow id="op1" input_data="d1">
 <data_processing_appl id="op2" input_data="d1">
  <data_processing_impl name="AttributeSelection" version=
      "1.7" libname="weka"/>
  <function_appl role="feature_subset_evaluator">
   <function_impl name="CfsSubsetEval" version="1.26" />
  </function_appl>
  <search_algorithm_appl role="search_algorithm">
   <search_algorithm_impl name="BestFirst" version="1.2"/>
    <parameter_setting name="D" value="1"/>
    <parameter_setting name="N" value="5"/>
  </search_algorithm_appl>
 </data_processing_appl>
</data_processing_workflow>
<dataset id="d2" name="kr-vs-kp-AttrSelection-
    CfsSubsetEval-BestFirst-D1-N5" url="http://..."
    output_of="op1,op2">
 <target_feature name="class" index="-1"/>
</dataset>
```

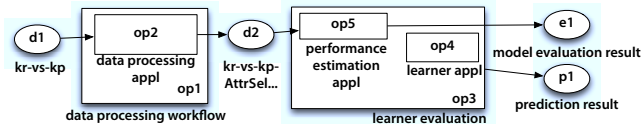


Figure 9. Data processing workflow in Exposé and ExpML

hence the inclusion of a `function appl` element with that role in the XML code. In turn, it has also a participant: a `function implementation`.

The second is a `search algorithm appl` that also has some parameter settings. Note that we make a small exception here: normally, a `parameter impl` subelement should be included in the `parameter setting`. Still, as an algorithm will only use its own parameter implementations, we chose for a more readable representation, in which simply the name is added as an attribute.

4.4 Experiment Evaluation and Results

As shown in the second half of the workflow at the bottom of Figure 9, the generated dataset serves as the input for a learner evaluation, which will in turn produce a model evaluation result and a prediction result.

The ExpML description is shown in Figure 10, and the corresponding ontological classes in Figure 4. It consists of a learning algorithm application complete with parameter settings and a base learner application with its own parameter settings. It also has a performance estimation technique (10-fold cross-validation) and a list of evaluation functions to assess the produced models, each pointing to their precise implementations. This level of detail is important to assess whether the results of this evaluation can be confidently reused. Indeed, there are many pitfalls associated with the statistical evaluation of algorithms [23, 8, 9]. By stating exactly which techniques were used, we can query for appropriate results. For instance, when comparing algorithms with cross-validation, it is important that the

```
<learner_evaluation id="op3" input_data="d2" series="exp1"
     experiment_id="445080">
 <learner_appl id="op4">
  <learner_impl name="weka.Bagging" version="1.31.2.2"
       libname="weka"/>
  <parameter_setting name="P" value="100"/>
  <parameter_setting name="I" value="10"/>
  <parameter_setting name="S" value="1"/>
  <learner_appl role="base-learner">
    <learner_impl name="weka.REPTree" version="1.19.2.2"
         libname="weka"/>
    <parameter_setting name="M" value="2"/>
    <parameter_setting name="V" value="0.0010"/>
    <parameter_setting name="N" value="3"/>
    <parameter_setting name="S" value="1"/>
    <parameter_setting name="L" value="-1"/>
  </learner_appl>
 </learner_appl>
 <performance_estimation_appl id="op5" input_data="d2">
  <performance_estimation_impl name="weka.
       crossValidateModel" version="1.53" libname="weka"/>
  <parameter_setting name="numfolds" value="10"/>
  <parameter_setting name="random" value="1"/>
 </performance_estimation_appl>
 <model_evaluation_function_appl name="predictive_accuracy
      ">
  <model_evaluation_function_impl name="weka.pctCorrect"
       version="1.53"  libname="weka"/>
 </model_evaluation_function_appl>
 ...
</learner_evaluation>
<model_evaluation_result  id="e1" output_of="op3,op5">
 <machine>vic_ni-09-10</machine>
 <evaluation name="build_cputime" value="2.25"/>
 <evaluation name="build_memory" value="36922896"/
 <evaluation name="mean_absolute_error" value="0.017"/>
 <evaluation name="root_mean_squared_error" value="0.085"
      />
 <evaluation name="predictive_accuracy" value="0.991"/>
 <evaluation name="confusion_matrix" value="[[won,nowin
      ],[1660,19],[9,1508]]"/>
 <evaluation name="precision_array" value="[[won,nowin
      ],[0.988,0.994]]"/>
 ...
</model_evaluation_result>
<prediction_result id="p1" output_of="op3">
  <prediction instance="0000" value="won">
    <probability outcome="won" value="0.985"/>
    <probability outcome="nowin" value="0.015"/>
  </prediction>
  ...
  <prediction instance="3195" value="nowin">
    <probability outcome="won" value="0.010"/>
    <probability outcome="nowin" value="0.990"/>
  </prediction>
</prediction_result>
```

Figure 10. An experiment workflow (setup and results) in ExpML.

same folds are used for all algorithms.

The output of the experiment is shown next, consisting of all evaluation results (also stating the machine used in order to interpret cpu time) and all predictions, including the probabilities for each class. Although omitted for brevity, evaluation error margins are stored as well. Storing predictions is especially useful if we want to apply new evaluation metrics afterwards without rerunning all prior experiments. We do not provide a format to describe models, as there already exist such formats (e.g. PMML).

5 Organizing Machine Learning Information

With thousands of ExpML files detailing performed experiments, the final step is to organize all this information in searchable databases so that it can be retrieved, rearranged, and reused in further studies. Although it is outside of the scope of this paper, we use the same ontological domain model to define a relational database model, so that future refinements of the domain model can be translated more easily to refinements of the database. This leads to a very fine-grained database model, warranting detailed queries about many different aspects of machine learning experiments. Currently, the database stores about 650,000 experiments on 54 classification algorithms, 87 datasets and 45 evaluation metrics. Simple data processing workflows, including feature selection and learning curve analyses are also used. All details can be found on the ExpDB website, at `http://expdb.cs.kuleuven.be`.

6 Meta-learning studies

We now illustrate the use of such databases for meta-learning investigations, increasingly making use of the available meta-level descriptions of algorithms and datasets. In each case, we use a single database query to generate the results. While we won't show the queries here, they can again be found on the ExpDB website. Most of these illustrations were presented before in Vanschoren et al. [28], and additional examples can be found there as well.

6.1 Ranking Algorithms

When selecting interesting learning approaches, or when testing the effectiveness of a new algorithm, one is often interested in a ranking of the available methods. To investigate whether some algorithms consistently rank high over various problems, we can query for their average rank (using each algorithm's optimal observed performance) over a large number of datasets, using optimized parameter settings. Figure 11 shows such a ranking over all UCI datasets.[9] To check which algorithms perform significantly different over many datasets, we used the Friedman test [8]. The right axis shows the average rank divided by the *critical difference*, meaning that two algorithms perform significantly different if the average ranks of two algorithms differ by at least that value (one unit).[10]

This immediately shows that indeed, some algorithms rank higher on average than others on the UCI datasets. Bagged naive Bayes trees seem to come in first overall, but the difference is not significant compared to that of SVMs with a polynomial kernel (although this SVM seems to outperform C4.5). Also note that bagging and boosting PART and NBTrees seem to yield big performance boosts, while boosting random trees proves particularly ineffective.

6.2 Tuning Parameters

By querying for the values of the parameter settings in each experiment, we can investigate their effects on several kinds of data. For instance, Figure 12 shows the effect of the kernel width (gamma) of the RBF kernel on a number of different datasets with the same default accuracy (10%). Note that on some datasets, increasing the gamma value will steeply increase performance, until it reaches a maximum and then slowly declines, while on other datasets, performance immediately starts decreasing up to a point, after which it quickly drops to default accuracy. Querying for the number of attributes in each dataset (shown in brackets), suggests that only datasets with few attributes benefit from large gamma values. Further investigation showed that the used SVM implementation easily starts overfitting when both gamma and the number of attributes are high [28]. This is an example of a query suggesting ways to improve an algorithm.

[9] We selected 18 algorithms to limit the amount of statistical error generated by using 'only' 87 datasets.

[10] The critical difference was calculated using the Nemenyi test with p=0.1, 18 algorithms and 87 datasets.

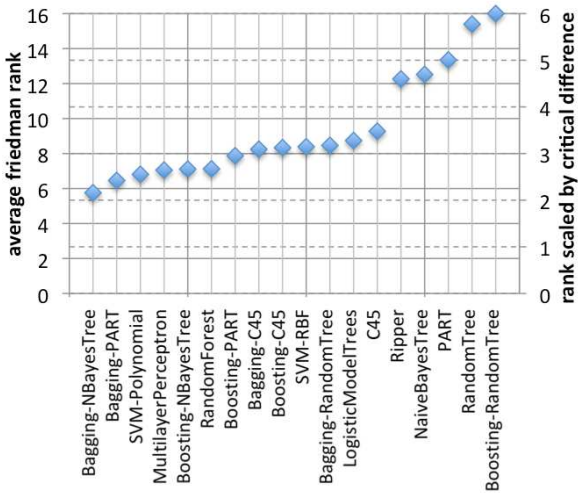


Figure 11. Average rank, specific algorithm setups.

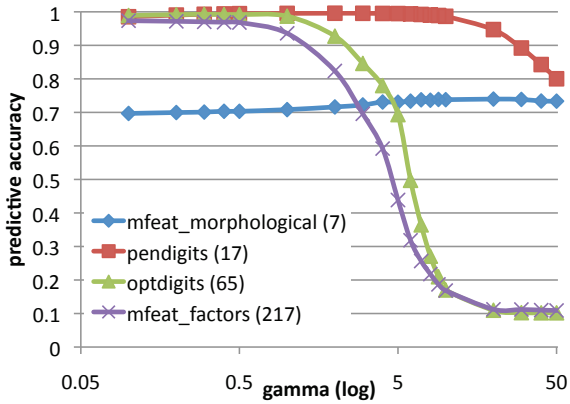


Figure 12. Effect of the RBF kernel width on different datasets.

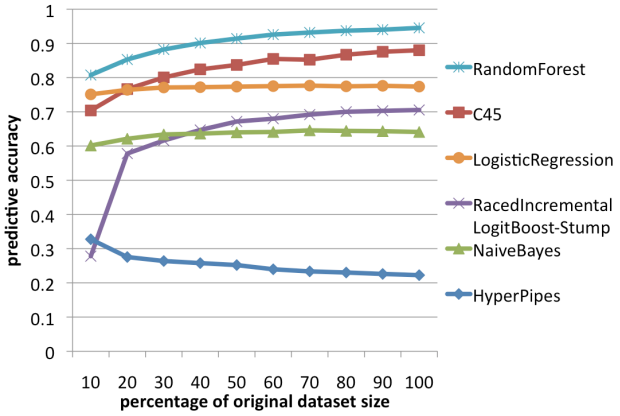


Figure 13. Learning curves on the Letter-dataset.

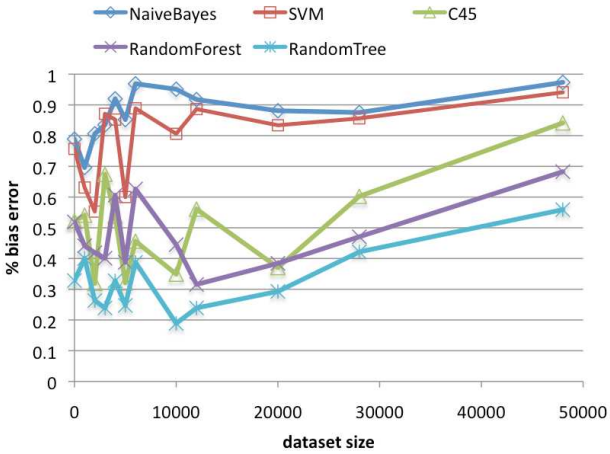


Figure 14. The average percentage of bias-related error in algorithms as a function of dataset size.

6.3 Applying Preprocessors

In many cases, data needs to be preprocessed before a certain learning algorithm can be used effectively. Since the database stores the entire experiment workflow, we can analyze the effect of preprocessing techniques on the performance of learning algorithms. For instance, to investigate how much data a certain algorithm needs to perform optimally, we can query for the results on downsampled versions of the dataset, yielding a learning curve for each algorithm, as shown in Figure 13. Note that some learning curves cross, indicating that the ranking of those algorithms depends on the size of the sample. While logistic regression is initially stronger than C45, the latter keeps on improving when given more data. However, to investigate the effect of preprocessing methods, the database model will need to be extended further.

6.4 Generalizing over Algorithms

We may also want to investigate which general properties of an algorithm might be desirable when approaching a certain task. One very interesting property of an algorithm is its bias-variance profile [14]. Since the database contains a large number of bias-variance decomposition experiments, we can give a realistic numerical assessment of how capable each algorithm is in reducing bias and variance error. The (maximum) percentage of bias-related error, compared to the total error, is stored in the database as an algorithm property.

We can also query for the effect of the dataset size on the dominance of the bias error component. Averaging the bias-variance results over datasets of similar size for each algorithm produces the result shown in Figure 14. It shows that bias error is of varying significance on small datasets, but steadily increases in importance on larger datasets, for all algorithms. For large datasets, it is thus advisable to choose a low-bias algorithm. This corroborates a previous study [4], but now on a much larger collection of datasets.

6.5 Mining for Patterns

Finally, instead of studying different dataset properties independently, we could also use data mining techniques to find patterns in the behavior of algorithm on various kinds of data. When querying for the default performance of OneR and J48 (WEKA's C4.5 implementation) on all UCI datasets, and plotting them against each other, we obtain Figure 15. It shows that on a large number of datasets, their performance is indeed about the same. Still, J48 works much better on many other datasets.

To automatically learn under which conditions J48 clearly outperforms OneR, we queried for the characteristics of each dataset, and discretized the data into three class values: "draw", "win_J48" (>4%

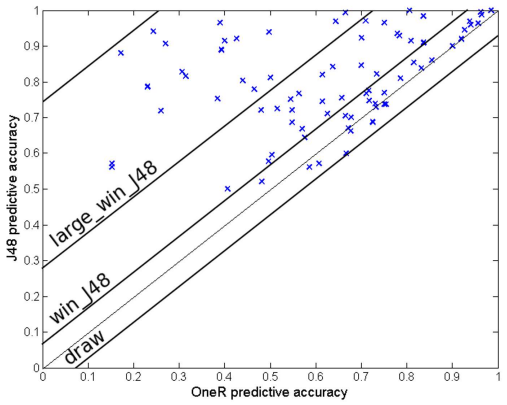


Figure 15. Performance comparison of J48 and OneR on all UCI datasets.

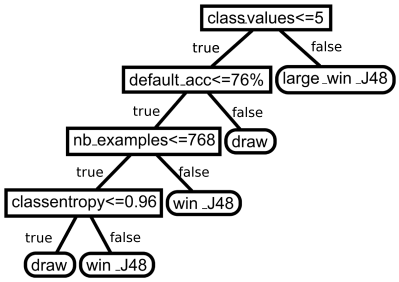


Figure 16. Meta-decision tree showing when J48 is better than OneR

gain), and "large_win_J48" (>20% gain). The tree returned by J48 on this meta-dataset is shown in Figure 16, showing that a high number of class values often leads to a large win of J48 over 1R.

Many more interesting meta-models could be discovered by simply querying the database and mining the results.

7 Conclusions

In this paper, we take an important step towards an infrastructure for collaborative experimentation in machine learning, in which experiments from many researchers can be automatically shared, organized, and analyzed in depth simply by querying experiment repositories. First, we describe the most important aspects of Exposé, a proposed ontology for machine learning experimentation. Next, we illustrate how we can translate this ontology into a formal XML-based language, ExpML, to describe experiments. Finally, we use an experiment database, also modeled after Exposé and filled with large amounts of classification experiments, to perform various in-depth meta-learning studies.

REFERENCES

[1] Aha, D.: Generalizing from case studies: A case study. Proceedings of the 9th International Conference on Machine Learning pp. 1–10 (1992)

[2] Ali, S., Smith, K.: On learning algorithm selection for classification. Applied Soft Computing Journal **6**(2), 119–138 (2006)

[3] Blockeel, H., Vanschoren, J.: Experiment databases: Towards an improved experimental methodology in machine learning. Lecture Notes in Computer Science **4702**, 6–17 (2007)

[4] Brain, D., Webb, G.: The need for low bias algorithms in classification learning from large data sets. PKDD '02: Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery pp. 62—73 (2002)

[5] Brazdil, P., Giraud-Carrier, C., Soares, C., Vilalta, R.: Metalearning: Applications to data mining. Springer (2009)

[6] Caruana, R., Niculescu-Mizil, A.: An empirical comparison of supervised learning algorithms. Proceedings of the 23rd International Conference on Machine Learning (ICML'06) pp. 161–168 (2006)

[7] Chandrasekaran, B., Josephson, J.: What are ontologies, and why do we need them? IEEE Intelligent systems **14**(1), 20–26 (1999)

[8] Demsar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research **7**, 1–30 (2006)

[9] Dietterich, T.: Approximate statistical tests for comparing supervised classification learning algorithms. Neural computation **10**(7), 1895–1923 (1998)

[10] Hand, D.: Classifier technology and the illusion of progress. Statistical Science (2006)

[11] Hilario, M., Kalousis, A., Nguyen, P., Woznica, A.: A data mining ontology for algorithm selection and meta-mining. Proceedings of the ECML/PKDD09 Workshop on 3rd generation Data Mining (SoKD-09) pp. 76–87 (2009)

[12] Horridge, M., Knublauch, H., Rector, A., Stevens, R., Wroe, C.: A practical guide to building owl ontologies using protege 4 and co-ode tools. The University of Manchester (2009)

[13] Hoste, V., Daelemans, W.: Comparing learning approaches to coreference resolution. there is more to it than bias. Proceedings of the Workshop on Meta-Learning (ICML-2005) pp. 20–27 (2005)

[14] Kalousis, A., Hilario, M.: Building algorithm profiles for prior model selection in knowledge discovery systems. Engineering Intelligent Systems **8**(2) (2000)

[15] Karapiperis, S., Apostolou, D.: Consensus building in collaborative ontology engineering processes. Journal of Universal Knowledge Management **1**(3), 199–216 (2006)

[16] Keogh, E., Kasetty, S.: On the need for time series data mining benchmarks: A survey and empirical demonstration. Data Mining and Knowledge Discovery **7**(4), 349–371 (2003)

[17] Kuehl, R.: Design of experiments: statistical principles of research design and analysis. Duxbury Press (1999)

[18] Michie, D., Spiegelhalter, D., Taylor, C.: Machine learning, neural and statistical classification. Ellis Horwood (1994)

[19] Nielsen, M.: The future of science: Building a better collective memory. APS Physics **17**(10) (2008)

[20] Panov, P., Soldatova, L., Dzeroski, S.: Towards an ontology of data mining investigations. Lecture Notes in Artificial Intelligence **5808**, 257–271 (2009)

[21] Perlich, C., Provost, F., Simonoff, J.: Tree induction vs. logistic regression: A learning-curve analysis. The Journal of Machine Learning Research **4**, 211–255 (2003)

[22] Pfahringer, B., Bensusan, H., Giraud-Carrier, C.: Meta-learning by landmarking various learning algorithms. Proceedings of the 17th International Conference on Machine Learning pp. 743–750 (2000)

[23] Salzberg, S.: On comparing classifiers: A critique of current research and methods. Data mining and knowledge discovery **1**, 1–12 (1999)

[24] Soldatova, L., King, R.: An ontology of scientific experiments. Journal of the Royal Society Interface **3**(11), 795–803 (2006)

[25] Stoeckert, C., Causton, H., Ball, C.: Microarray databases: standards and ontologies. nature genetics **32**, 469–473 (2002)

[26] Szalay, A., Gray, J.: The world-wide telescope. Science **293**, 2037–2040 (2001)

[27] Vanschoren, J., Blockeel, H., Pfahringer, B.: Experiment databases: Creating a new platform for meta-learning research. Proceedings of the ICML/UAI/COLT Planning to Learn Workshop, pp. 10–15 (2008)

[28] Vanschoren, J., Pfahringer, B., Holmes, G.: Learning from the past with experiment databases. Lecture Notes in Artificial Intelligence **5351**, 485–492 (2008)